2019

Physics-Guided Deep Learning for Power System Sate Estimation

Lei Wang
University of Central Florida

PHYSICS-GUIDED DEEP LEARNING FOR POWER SYSTEM STATE ESTIMATION

by

LEI WANG
B.S. Sichuan University, China, 2008
M.S. Sichuan University, China, 2012

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Science
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2019

ABSTRACT

Conventionally, physics-based models are used for power system state estimation, including Weighted Least Square (WLS) or Weighted Absolute Value (WLAV). These models typically consider a single snapshot of the system without capturing temporal correlations of system states. In this thesis, a Physics-Guided Deep Learning (PGDL) method incorporating the physical power system model with the deep learning is proposed to improve the performance of power system state estimation. Specifically, inspired by Autoencoders, deep neural networks (DNNs) are utilized to learn the temporal correlations of power system states. The estimated system states are checked against the physics law by a set of power flow equations. Hence, the proposed PGDL approach is both data-driven and physics-based. The proposed method is compared with the traditional methods on the basis of accuracy and robustness in IEEE standard cases. The results indicate that PGDL framework provides more accurate and robust estimation for power system state estimation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Modern electric grid is a complex and highly interconnected system. As an incredibly important system, the operating states should be constantly monitored, which is performed by a state estimator in an energy management system (EMS). It receives raw measurements from a Supervisory Control and Data Acquisition (SCADA) system. Generally, these measurements consist of voltage magnitudes, real and reactive power injections at buses and real and reactive power flows in lines. Given these measurements, state estimator generates reliable estimates of current system states which are critical inputs for other applications, such as optimal power flow and contingency analysis. Thus, an accurate and robust state estimator is vital in EMS. However, the challenge of power system state estimation is that the measurements acquired from SCADA system are subject to noise caused by the instrument or communication, or even gross errors in some cases.

To solve the problem, Fred C. Schweppe et al. proposed static-state estimation in 1970 [1, 2]. The algorithm they proposed uses the redundancy of measurements to filter the errors and achieve a reasonable estimation of system states. As a highly non-linear problem, state estimation commonly achieves the optimal solution by iteration. The Phasor Measurement Unit which is a new technique of measuring bus voltage and power flow is able to linearize the state estimation problem and hence, solve the problem without iteration [3]. Nevertheless, PMUs are not extensively spread in power grid. Therefore, state estimation keeps non-linear in this thesis.

The commonly used methods for power system state estimation are Weighted Least Square (WLS) and Weighted Least Absolute Value (WLAV). WLS is an iterative algorithm to estimate the states by minimizing the square of difference between actual measurements and estimated measurements. It is widely adopted in industrial field thanks to its computational efficiency. However, it is not robust to outliers (bad data). When there are bad data in the measurements, based on which the

estimated states of WLS can significantly deviate from the true states. Therefore, a post-processing procedure is required to identify and remove the bad data, which is extremely time-consuming. As for the WLAV, it overcomes the drawback of WLS and is robust to outliers by minimizing the L_1 norm between actual measurements and estimated measurements. WLAV can be solve by standard techniques such as Interior Point and Simplex. However, the shortcoming of WLAV is the computational efficiency. It takes longer time than WLS to solve the state estimation problem. Moreover, it fails when there are error present in leverage measurements [4].

In this paper, a framework termed as Physics-Guided Deep Learning (PGDL) is proposed to explore the possibility of applying deep learning in power system state estimation. The past decade has seen dramatic progress in the field of machine learning [5]. Deep learning has been applied in computer vision, speech recognition, natural language process, and many other areas [6]. As a notable example, AlphoGo developed by Google Deepmind employs deep reinforcement leaning and became the first computer program to defeat a professional human Go player [7]. With these achievements in computer science, the question arises on how to bridge machine learning with power systems.

In the power industry, machine learning has been used for load forecasting [8, 9]. In such application where the underlying physical models are unknown, machine learning presents a powerful tool to perform predictive analytics. However, most applications in power system analysis have concrete physical models (e.g. power flow equations) and machine learning may fall short compared to model-based approaches.

The proposed PGDL state estimator explores the temporal correlations between system states. In contrast to black-box modeling, the proposed PGDL is both data-driven and first-principle-based. Specifically, the true dynamics of power system state are unknown and difficult to model due to exogenous factors such as weather and customer behaviors. Therefore, we adopt the latest

development in machine learning and apply deep neural networks to learn the state dynamics and correlations. The estimated system states are checked against physics law by running through a set of power flow equations. The proposed PGDL state estimation method has been tested in standard IEEE cases. The performance improvement is clearly seen from the numerical results.

This thesis is organized as follows: Chapter 2 presents an overview of WLS and WLAV methods. Chapter 3 presents the machine learning framework and the proposed PGDL model for state estimation. In Chapter 4, the performance of PGDL model is evaluated in IEEE 14-bus and IEEE 118-bus systems. Chapter 5 discusses the findings from our study and provides concluding remarks.

CHAPTER 2: FORMULATIONS OF STATE ESTIMATION

In this chapter, an overview of WLS and WLAV in power system state estimation is presented, including the formulations and solution techniques.

Introduction

Power system state estimation is such an application backed by detailed physical models. In order to identify the current system states, state estimators provide reliable estimate using measurements collected from SCADA system [1]. The measurement data usually consist of real and reactive power injections, real and reactive power flows, and bus voltage magnitudes, while the state vector contains the voltage magnitudes and phase angles at all buses. Due to instrument errors and communication noises, a stochastic error term needs to be incorporated. State estimation aims to obtain true states by maximizing the likelihood of estimated states, equivalent to minimizing the residuals between estimated and actual measurements.

Specifically, the physical model of power system can be represented by

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \varepsilon \tag{2.1}$$

where $h(\cdot)$ is the non-linear functions relating measurements $\mathbf{z}$ to states $\mathbf{x}$, and ε denotes the measurements error vector [4]. Given redundant measurements $\mathbf{z}$ to estimate the states $\mathbf{x}$, it can be

expressed as an optimization problem shown as below:

$$\begin{aligned} \min_{x} \quad & J(z - h(x)) \\ \text{s.t.} \quad & g(x) = 0 \\ & c(x) \leq 0 \end{aligned} \tag{2.2}$$

where $J(\cdot)$ is the objective function, and $g(\cdot)$ and $c(\cdot)$ are constraints. The difference between WLS and WLAV is the objective function $J(\cdot)$.

Weighted Least Square

WLS-based state estimation will determine the optimal states by minimizing the weighted residuals:

$$\min_{\hat{\mathbf{x}}} \quad J(x) = [\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}})]^{\mathbf{T}} \mathbf{R}^{-\mathbf{1}} [\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}})] \tag{2.3}$$

where R is the measurement covariance matrix, and R^{-1} represents the weight matrix which indicates the accuracy of particular measurement. At the minimum of (2.3), the first-order optimality conditions are satisfied, which can formulated as:

$$g(x) = \frac{\partial J(x)}{\partial x} = -H^T(x) R^{-1} [z - h(x)] = 0 \tag{2.4}$$

where $H(x)$ is the Jacobian matrix of $h(\cdot)$. The solution of highly non-linear problem (2.3) can be obtained by solving (2.4), which usually can be solved by an iterative technique known as Gauss-Newton method.

Gauss-Newton algorithm for WLS

Typically, Gauss-Newton algorithm is adopted to solve the non-linear WLS problem. Expanding the function $g(\cdot)$ into Taylor series around the current solution and neglecting the higher order terms gives:

$$g(x) = g(x_i) + \frac{\partial g(x_i)}{\partial x}(x - x_i) = 0 \tag{2.5}$$

The solution of (2.5) is:

$$x_{i+1} = x_i - [G(x_i)]^{-1} g(x_i) \tag{2.6}$$

where i is the iteration index, x_i is the solution vector at i_{th} iteration, $G(x_i) = \frac{\partial g(x_i)}{\partial x}$ is called the Gain Matrix. This can be solved by iterative updates of x in (2.6) until a certain tolerance is achieved.

The iterative Gauss-Newton method for WLS-based state estimation can be outlined as follows:

1. Initialize the state vector x_0 (usually, a flat start) and set iteration index 0.
2. Calculate $h(x_i), H(x_i)$.
3. Calculate $G(x_i)$ using the results of step 2.
4. Solve (2.6) to obtain x_i
5. Test for convergence. If $x_{i+1} - x_i$ is smaller than the tolerance, stop. Else, update x and iteration counter and go to step 2.

Weighted Least Absolute Value

Taking advantage of the inherent ability of Least Absolute Value (LAV), the WLAV method of state estimation is developed as a more robust way to outliers. It tries to estimate the states by minimizing L_1 norm between the actual measurements and the estimated measurements. The

objective function of WLAV can be expressed as:

$$\begin{aligned} \min_{x} \quad & J(x) = \sum_{j=1}^{m} W_j \left| r_j \right| \\ \text{s.t.} \quad & r = z - h(x) \end{aligned} \tag{2.7}$$

where r is measurement residual vector and W_j is the reciprocal of the error variance for j^{th} measurement. The states obtained by solving (2.7) will strictly satisfy a set of selected measurements (the number of measurements is equal to the total number of states). The remaining measurements will be filtered out with their nonzero residuals. Therefore, WLAV is more robust to outliers relative to WLS counterpart.

The WLAV estimation problem (2.7) can be formulated as a linear programming (LP) problem, and solved by the well-developed LP methods. Let ξ be defined:

$$|r_i| \leq \xi, \quad 1 \leq i \leq m$$

Now, this inequality can be changed to two equalities by introducing two non-negative slack variables l_i and k_i:

$$r_i - l_i = -\xi_i$$

$$r_i + k_i = \xi_i$$

Let $u_i = \frac{1}{2} l_i$ and $v_i = \frac{1}{2} k_i$, then:

$$u_i - v_i = r_i$$

$$u_i + v_i = \xi_i$$

We know that (2.1) can be linearized as:

$$H_x \Delta x + \xi = \Delta z \tag{2.8}$$

where H_x is the Jacobian Matrix at point x. Let Δx^u and Δx^l defined such that they are non-negative and $\Delta x = \Delta x^u - \Delta x^l$. Then, (2.7) can be rewritten in terms of new variables:

$$\begin{aligned} \min_{x} \quad & J(x) = \sum_{j=1}^{m} W_j(u_i + v_i) \\ \text{s.t.} \quad & h(\Delta x^u - \Delta x^l) + u - v = \Delta z \\ & u, v, \Delta x^u, \Delta x^l \geq 0 \end{aligned} \tag{2.9}$$

The WLAV-based state estimation is solved by obtaining the solutions of LP problem (2.9). The equivalence between WLAV and LP was presented in [10]. The LP problem (2.9) can be solved using well-developed methods such as Simplex or Interior Point(IP). In this thesis, IP solver [11] will be employed to solve the WLAV state estimation problem.

CHAPTER 3: PHYSICS-GUIDED DEEP LEARNING FRAMEWORK

Typically, power system conditions are constantly changing due to weather, customer behaviors, and other events. Therefore, system states are also correlated from time to time, i.e., temporal correlations exist between states at different times. Nevertheless, the state dynamics and correlations are very difficult to explicitly model because the underlying physics is unknown with random behaviors involved. Therefore, the first-principal-based single-snapshot state estimation is not capable of capturing the full dynamics of power system.

In this thesis, a PGDL state estimator is proposed to explore the temporal correlations between system states. In contrast to black-box modeling, the proposed PGDL is both data-driven and first-principle-based. Specifically, the true dynamics of power system state are unknown and difficult to model due to exogenous factors such as weather and customer behaviors. Therefore, latest development in machine learning and deep neural networks are employed to learn the state dynamics and correlations. The estimated system states are checked against physics law by running through a set of power flow equations.

Machine Learning Framework

The motivation behind our study is that machine learning may be able to learn temporal correlations among different states. The proposed deep-learning based framework is described in Fig. 3.1.

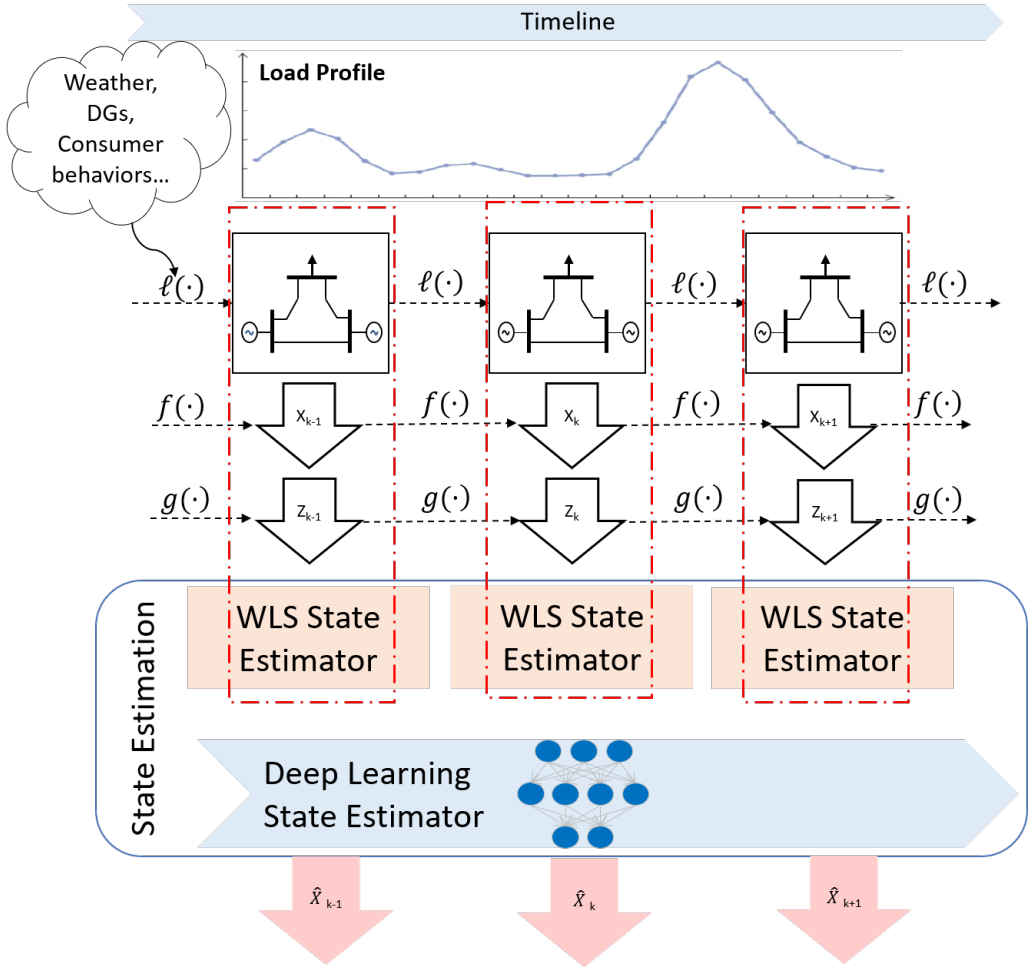

Figure 3.1: Deep-Learning based framework to capture state correlations

In a power system, load is constantly changing due to underlying causes such as weather and customer behaviors. Assume that the load at time t ($\mathbf{d_t}$) is correlated with load at time $t-1$ ($\mathbf{d_{t-1}}$), denoted by $\mathbf{d_t} = \ell(\mathbf{d_{t-1}})$. This load correlation is translated to state correlation through the complex operations of the power system to balance load and generation. The state correlation is represented by $\mathbf{x_t} = \mathbf{f}(\mathbf{x_{t-1}})$, where $f(\cdot)$ denotes the nonlinear nature of power system operations. The correlations among state variables are then translated to measurement correlations through the nonlinear physical power flow model, denoted by $\mathbf{z_t} = \mathbf{g}(\mathbf{z_{t-1}})$. The temporal correlations of

measurement data should be considered when estimating states, whereas a single-snapshot state estimation lacks this consideration.

If we consider state estimation as an optimization problem, the traditional method is formulated as in Eq. (2.3), which minimizes the estimation errors for one time shot. On the contrary, the state estimation with temporal correlations is formulated as a look-ahead optimization problem:

$$\begin{aligned} \min_{\hat{\mathbf{x}}_\mathbf{t}} \quad & \sum_t [\mathbf{z_t} - \mathbf{h}(\hat{\mathbf{x}}_\mathbf{t})]^\mathbf{T} \mathbf{R}^{-\mathbf{1}} [\mathbf{z_t} - \mathbf{h}(\hat{\mathbf{x}}_\mathbf{t})] \\ \text{s.t.} \quad & \hat{\mathbf{x}}_\mathbf{t} = \mathbf{f}(\hat{\mathbf{x}}_{\mathbf{t-1}}) \end{aligned} \tag{3.1}$$

Note that the true dynamics of states $f(\cdot)$ is unknown and difficult to mathematically model. Therefore, the modeling of state correlations is challenging. This motivates a data-driven approach that learns the dynamics through historical data. A Deep Neural Network (DNN) with sufficient learning capacity is suitable for the large amount of data in a power system. The PGDL framework uses DNN to learn the temporal correlations of state variables while incorporating the physical model.

Physics-Guided Deep Learning

The PGDL model that incorporates historical data and the physical power flow model is proposed in this section. This hybrid learning method employs deep neural networks to learn the state correlations, yet considers physical flow in a power system.

Physics-Guided Learning Architecture

A hybrid machine learning model is proposed inspired by the emerging Autoencoder in the Artificial Intelligent (AI) field . As shown in Fig. 3.2, an Autoencoder is a neural network that is

trained to copy its input to its output [6]. Internally, it has a hidden layer that divides the neural network into two parts: encoder and decoder. The Autoencoder is initially designed for dimension reduction so that the number of features can be reduced to represent a system. To apply Autoencoders in state estimation, the measurement $\mathbf{z}$ is fed into the encoder, and the hidden layer output is the estimated system states $\hat{\mathbf{x}}$, which can be considered as features to represent the power grid. The estimated states $\hat{\mathbf{x}}$ then go through the decoder to output the reconstructed measurement data $\hat{\mathbf{z}}$. In the context of power system state estimation, measurement $\mathbf{z}$ come from the SCADA system consisting of real and reactive power injection at each bus (P_i and Q_i for bus i), and real and reactive power flow on each transmission line (P_{ij} and Q_{ij} for line ij). Estimated values of these measurements are denoted by $\hat{\mathbf{z}}$. System states $\mathbf{x}$ consists of voltage magnitude and angle at each bus (V_i and θ_i for bus i). The corresponding estimated system states are denoted by $\hat{\mathbf{x}}$.

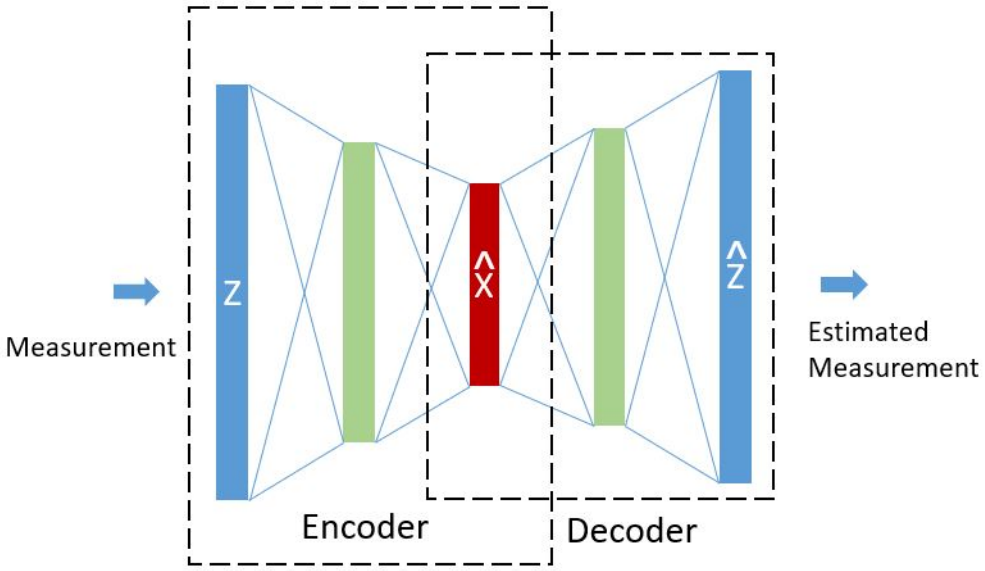

Figure 3.2: Autoencoder for state estimation

Using Autoencoder for state estimation would be a pure data-driven method without explanatory

models. However, with the domain knowledge of power systems, we could improve the Autoencoder by incorporating the first principles in the power grid. The proposed physics-guided learning is both data-driven and first-principle-based.

Fig. 3.3 depicts the physics-guided learning architecture. At time k, the DNN input is the measurement set $\mathbf{z^k} = [z_1^k, z_2^k, \cdots, z_m^k]^T$ and the output is the corresponding estimated state vector $\hat{\mathbf{x}}^\mathbf{k} = [\hat{x}_1^k, \hat{x}_2^k, \cdots, \hat{x}_n^k]^T$ (m: the number of measurements, n: the number of states). The state vector comprises voltage magnitude and angle at each bus.

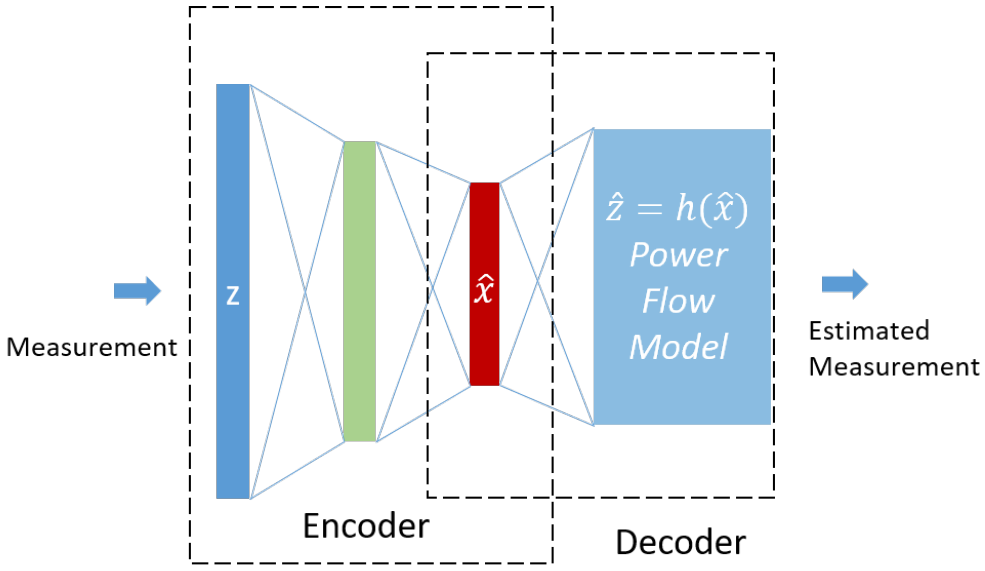

Figure 3.3: Hybrid Learning Structure

The state estimation then will go through the physical measurement model consisting power flow equations (3.2). As a result, estimated measurement vector $\hat{\mathbf{z}}$ is generated. To train the DNN, the loss function is the cumulative differences between actual and estimated measurement vectors. This error is back-propagated through the physical model layer and the neural network layers, so

that weights and biases of the neural net will be adjusted accordingly.

$$
\begin{aligned}
P_i &= \sum_{j=1}^{N} V_i V_j (G_{ij}\cos(\theta_i - \theta_j) + B_{ij}\sin(\theta_i - \theta_j)) \\
Q_i &= \sum_{j=1}^{N} V_i V_j (G_{ij}\sin(\theta_i - \theta_j) + B_{ij}\cos(\theta_i - \theta_j)) \\
P_{ij} &= -{V_i}^2 G_{ij} + V_i V_j (G_{ij}\cos(\theta_i - \theta_j) + B_{ij}\sin(\theta_i - \theta_j)) \\
Q_{ij} &= -{V_i}^2 B_{ij} + V_i V_j (G_{ij}\sin(\theta_i - \theta_j) - B_{ij}\cos(\theta_i - \theta_j))
\end{aligned}
\tag{3.2}
$$

where:

V_i is the voltage magnitude at bus i

P_i is the real power injection at bus i

Q_i is the reactive power injection at bus i

P_{ij} is the real power flow from bus i to bus j

Q_{ij} is the reactive power flow from bus i to bus j

G_{ij} is the real part in admittance matrix

B_{ij} is the imaginary part in admittance matrix

Data and Training Flow

The implementation details are described in Fig. 3.4. The PGDL model consists of multiple stages: data acquisition, initialization, pre-training, validation, and online estimation.

First, measurement data are acquired. In our case studies, the data come from simulations, but in reality, the data are acquired from the SCADA system. The data are then divided into the training set, validation set, and testing set. The training data are used to train the DNNs. For the pre-training stage, different learning rates and initialization methods are used. Note that the pre-training is an offline batch training, where the losses are calculated and back-propogated through

the neural networks for the update of the weights. After the DNNs are trained, validation data are used to assess the overall performance of the DNNs. If the performance is satisfied, go on to online estimation with new data coming in. Otherwise, go back and optimize the DNN parameters and repeat the process.

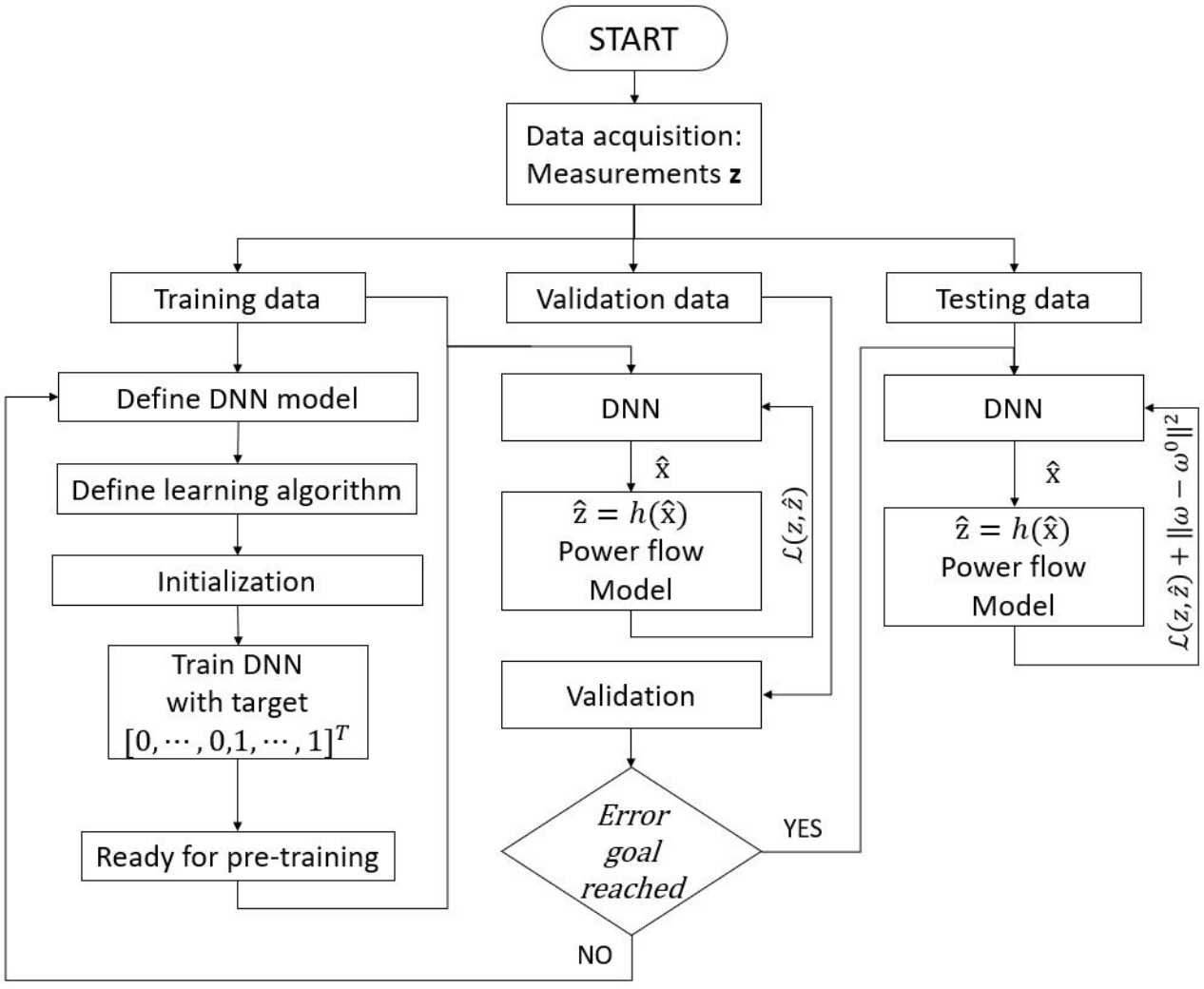

Figure 3.4: PGDL flowchart

Deep Neural Networks

Deep neural networks are used as the encoder to map the nonlinear relationship of measurement $\mathbf{z}$ and states $\mathbf{x}$. It is well-known that neural networks has the universal approximation capability to approximate nonlinear functions [12]. In this paper, two typical deep neural networks are studied: Feedfoward neural nets (FFNN) and Long short-term memory (LSTM) neural nets.

Feedforward Neural Nets

A multi-layer FFNN is first utilized to learn the true dynamics among system states. As shown in Fig. 3.5, the input layer has m neurons, where m is the number of elements in a measurement set. The width and depth of network will be determined by the complexity of the power system. The output layer has n neurons, where n is the number of estimated state variables. Usually, $n = 2 * N - 1$, N is the number of buses in the power system. For the hidden layers, the hyperbolic tangent ($tanh$) activation functions are exploited. It is proved that the hyperbolic tangent networks, unlike the $sigmoid$, do not suffer from the saturation behavior of the top hidden layers due to its symmetry around 0 [13].

Regarding the IEEE 14-bus and IEEE 118-bus systems used in this paper, the details of FFNN structure are shown in Table 3.1.

Table 3.1: Structures of FFNN in IEEE test cases

	input layer	hidden layers	output layer
	m	$l_1 * l_2 * \cdots * l_k$	n
IEEE14	30	128*128*64	27
IEEE118	330	512*512*512*512*512	235

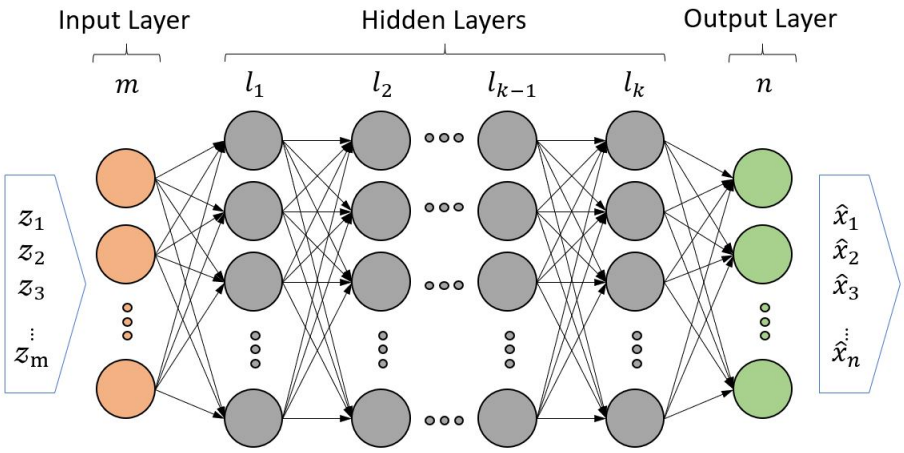

Figure 3.5: Feedforward neural network

Note that FFNN only takes the measurement at time t as input, and hence does not specifically learn the temporal correlations. FFNN is chosen as a comparing model to test the hypothesis that adding temporal learning capability in the encoder DNN would enhance the overall performance of system estimation. This hypothesis is verified in the numerical results.

LSTM Neural Nets

LSTM networks are Recurrent Neural Network (RNN) that is suitable to capture the state dynamics [14]. Unlike conventional RNNs, blueLSTM can alleviate the exploding and vanishing gradient problems due to the memory cell and gating mechanism [15]. A common LSTM unit comprised of a memory cell and three gates is described in Fig. 3.6.

For a single LSTM unit, the inputs are the measurement vector $\mathbf{z_t}$ at time t and hidden vector $\mathbf{h_{t-1}}$ from last time step, while the output is $\mathbf{h_t}$.There are three gates, input gate i, forget gate f, and output gate o. The LSTM unit will learn the dependencies between the data in the input sequence.

The input gate manages the values flowing into the memory cell from the inputs. The forget gate determines which part is passed to the next step. As for the output, it is a product of the result of output gate and the activation of the memory cell. The mathematical expressions of LSTM are given as follows:

$$f_t = \sigma(W_f z_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i z_t + U_i h_{t-1} + b_i)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ (tanh(W_c z_t + U_c h_{t-1} + b_c))$$

$$o_t = \sigma(W_o z_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t \circ tanh(c_t)$$

where Ws and Us are weights that connect different layers, while b_f, b_i, b_c, b_o are the biases with each individual gate. $\sigma(\cdot)$ represents the $sigmoid$ activation function, and $tanh$ is the hyperbolic tangent function.

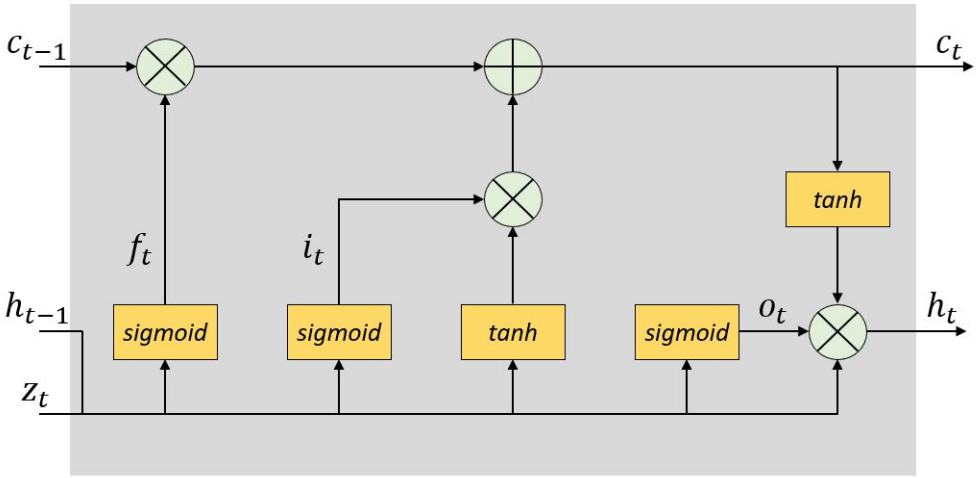

Figure 3.6: LSTM Structure

To improve the learning capability, multiple LSTM units are stacked as shown in Fig. 3.7 and the

last layer is a fully connected linear layer. Note that the input of LSTM is also the measurement at time t, but because of its recurrent nature, all previous time steps are implicitly considered. Fig. 3.7 shows an unrolled LSTM in time, where it is two dimensional in time and space.

For the IEEE cases in this paper, Table 3.2 provides the details of stacked LSTM network configuration.

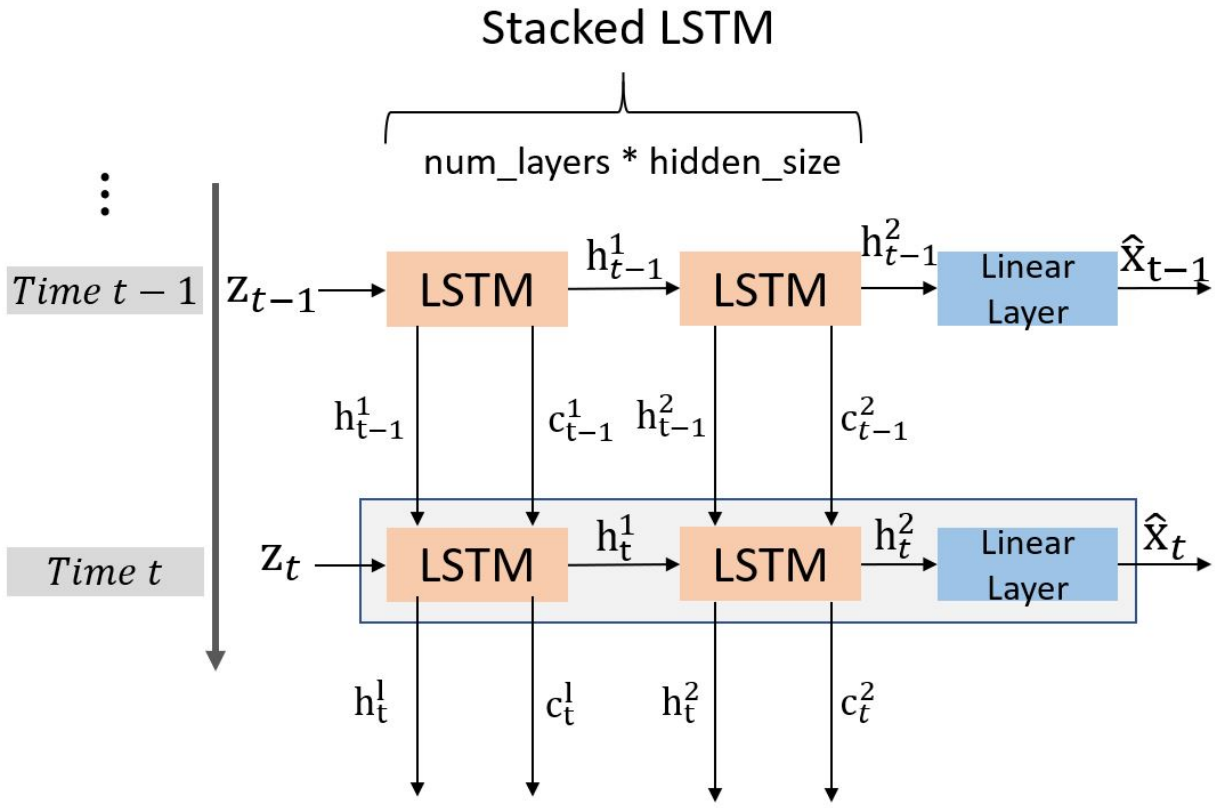

Figure 3.7: blueStacked LSTM unrolled in time

Table 3.2: Structures of stacked LSTM in IEEE test cases

	Stacked LSTM		Linear Layer
	input size	num_layers*hidden_size	
IEEE14	30	2*128	27
IEEE118	330	2*512	235

Training Procedure

The training procedure of two DNNs consists of multiple implementation steps including network configuration and initialization, defining loss functions and regularization, and selecting proper learning algorithms and learning rates.

The performance of DNNs can vary depending on the configuration and initialization. For each dataset, we run DNNs with 20 different configurations and random seeds, and select the DNNs with the best performance on the validation data, in order to achieve reliable online estimation for the test data. For the initialization, several methods are attempted including random initialization, uniform initialization, and $Xavier$ initialization [13]. The neural networks are then trained using target values of voltage angels and magnitudes equal to $[0, 0, \cdots, , 1, 1, \cdots, 1]^T$, which is similar to the $flat\ start$ in WLS state estimation.

The loss function needs to be defined to train and update weights and biases of DNNs. Cumulative mean square error (MSE) loss function is used as in equation blue(3.3).

$$\mathcal{L} = \frac{1}{m} \sum_{t} [\mathbf{z_t} - \hat{\mathbf{z}}_\mathbf{t}]^\mathbf{T} [\mathbf{z_t} - \hat{\mathbf{z}}_\mathbf{t}] \tag{3.3}$$

For online training and estimation, a regularization technique is implemented to maintain the stability of the trained DNNs. A $L^2 - SP$ penalty [16] is added to encourage similarity with the

$starting\ point$, i.e., the parameters of pre-trained model. The loss function with $L^2 - SP$ penalty can be formulated as (3.4).

$$\mathcal{L} = \frac{1}{m}[\mathbf{z} - \hat{\mathbf{z}}]^{\mathbf{T}}[\mathbf{z} - \hat{\mathbf{z}}] + \left\|\mathbf{w} - \mathbf{w^0}\right\| \tag{3.4}$$

where w^0 is the parameter vector of the pre-trained DNNs.

The loss is then backpropagated through the physical model layer as well as the neural net layer. Hence, the chain rule is expressed by:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{z}}} \frac{\partial \hat{\mathbf{z}}}{\partial \hat{\mathbf{x}}} \frac{\partial \hat{\mathbf{x}}}{\partial \mathbf{w}} \tag{3.5}$$

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}} \tag{3.6}$$

Here η is the learning rate. In this work, a constant learning rate, a decayed learning rate, and a cyclical learning rate are attempted to improve the performance [17]. The update of weights and biases is based on the gradient-descent algorithm [18]. Specifically, mini-batch gradient descent is employed to pre-train the DNN models. For online training and estimation, stochastic gradient descent is used to continuously update model parameters as new data points come in.

CHAPTER 4: NUMERICAL RESULTS

In this section, the proposed PGDL is implemented in two standard IEEE power systems. The performance is evaluated and compared with conventional WLS-based and WLAV-based state estimation in terms of accuracy and robustness. The performance measure is Mean Absolute Estimated Error (MAEE):

$$MAEE = \frac{1}{N}\sum_{k=1}^{N}|\hat{x}_k - x_k| \tag{4.1}$$

where N is the total number of buses, $\hat{x}_k$ and x_k are the estimated and real state values (voltage magnitude or voltage angle) at the k^{th} bus. The simulation is carried out in a predefined time period, and the mean and standard deviation of the MAEEs are compared.

Simulation Setup

The simulation data are generated in IEEE 14-bus and IEEE 118-bus systems. The load profile is downloaded from NYISO and scaled down for the two systems [19]. A one-day normalized load profile with 5-minute time interval is given in Fig. 4.1.

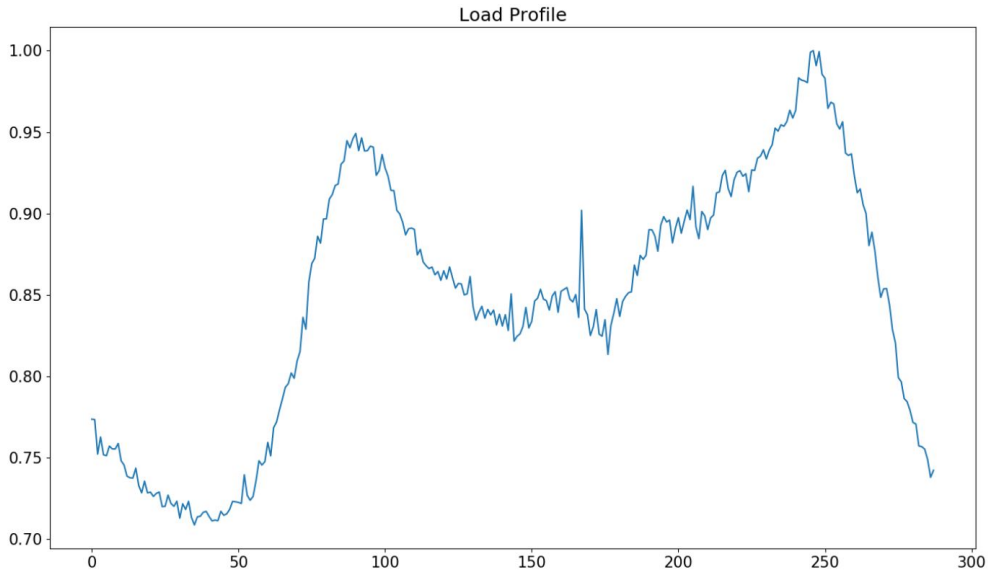

Figure 4.1: Normalized Load Profile from NYISO

The true states and measurement vectors are generated using MATPOWER 6.0 [20]. The states consist of voltage magnitude and angle of each bus, which are 27 and 235 for the 14-bus and 118-bus system respectively (the voltage angle of reference bus is 0).

After obtaining the measurement vectors $\bar{\mathbf{z}}$ including $[V, P_i, Q_i, P_{ij}, Q_{ij}]^T$ from MATPOWER, white Gaussian noises are added to form the noised measurements $\mathbf{z} = \bar{\mathbf{z}} + a\sigma$ where $a \sim \mathcal{N}(0, 1)$. The number of each measurement and standard deviation are shown in Table 4.1. Note that the number of measurements is greater than the number of states to ensure the observability [21].

Table 4.1: Measurement vectors of IEEE test systems

measurements	IEEE14	IEEE118	standard deviation σ_i
V	1	1	0.004
P_i	7	118	0.01
Q_i	5	118	0.01
P_{ij}	7	44	0.008
Q_{ij}	10	49	0.008

The time horizon of dataset is 4 months, i.e., 34560 data points with 5-minute time interval. The first two month with 17280 data points are split into two part: training (70%) and validation (30%). The training data are used in both initialization and pre-training stages. The validation data are used to validate the PGDL performance . The remaining two months data are used for online training and state estimation, and the performance of the last day (288 data points) is reported for comparative study.

Performance Analysis - Accuracy

The MAEEs for IEEE 14-bus system are presented in Figs. 4.2 and 4.3. It is observed that WLS has the largest variations of estimation errors. In contrast, FFNN and LSTM result in small error variations. In terms of accuracy, the two neural networks also have smaller average MAEEs. To compare the two neural net configurations, we found that while the average MAEEs are about the same, LSTM has lower standard deviation of errors than FFNN.

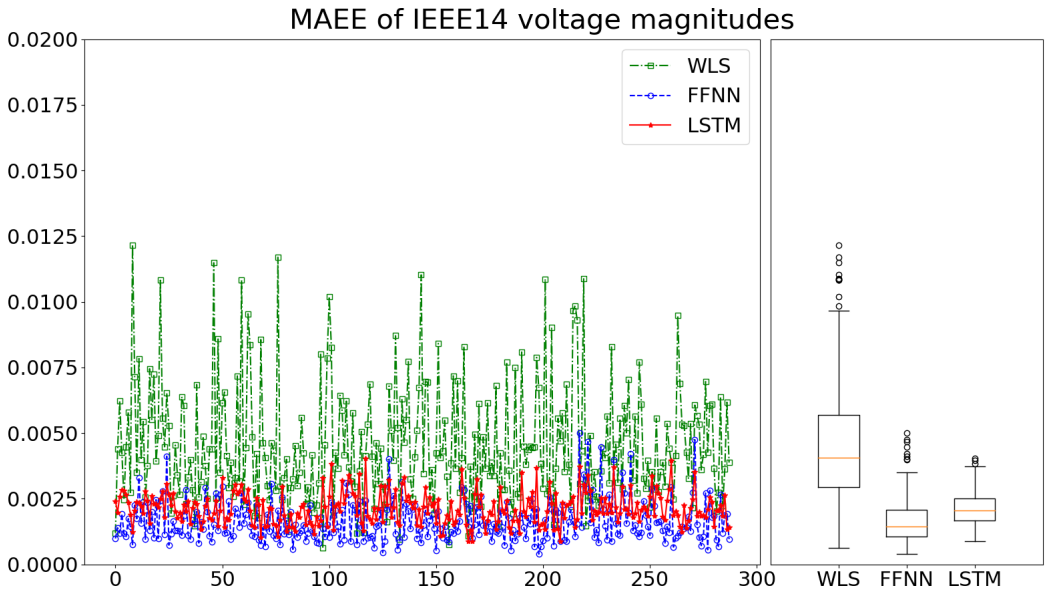

Figure 4.2: MAEEs of voltage magnitudes in IEEE 14-bus system

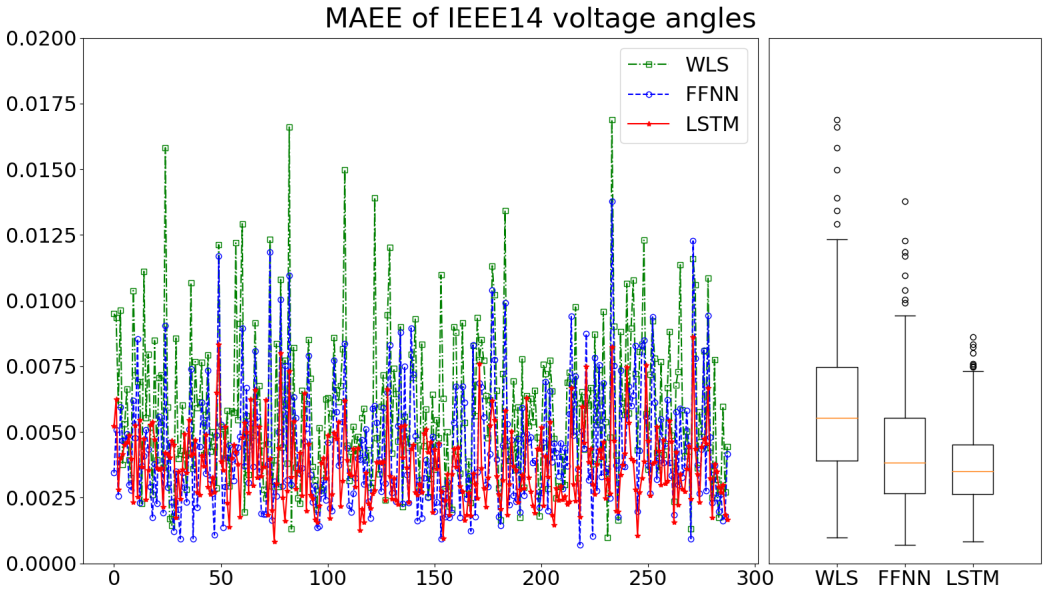

Figure 4.3: MAEEs of voltage angles in IEEE 14-bus system

For detailed analysis, the mean and standard deviation are described in Table 4.2. Generally, WLS has the highest average error for voltage magnitude and angle while two deep-learning based models are more accurate. FFNN and LSTM also have lower standard deviation of MAEE indicating more stable performance throughout the testing period. We also notice that the improvement of voltage magnitude estimation is more significant than voltage angle estimation.

Table 4.2: Mean and standard deviation of MAEEs in IEEE 14-bus system

IEEE14	Mean(MAEE)		SD(MAEE)	
	V	θ	V	θ
WLS	0.0046	0.0059	0.0022	0.0029
FFNN	0.0017	0.0044	0.0008	0.0023
LSTM	0.0021	0.0037	0.0006	0.0014

To generalize the proposed approach, IEEE 118-bus system is used to test the performance in a larger system. Figs 4.4 and 4.5 provide a graphical illustration of the MAEEs between WLS and two DNN-based methods. It is clearly seen that LSTM provides the best estimates for voltage magnitudes, while FFNN performs the best for voltage angles. Table 4.3 shows a statistical comparison of WLS and two DNN-based approaches. Overall, DNN-based methods outperform WLS. LSTM performs significantly better on voltage magnitude, while its performance of voltage angle estimation are on par with WLS. On the other hand, The DNN-based method are much more stable than WLS estimation with lower error variance.

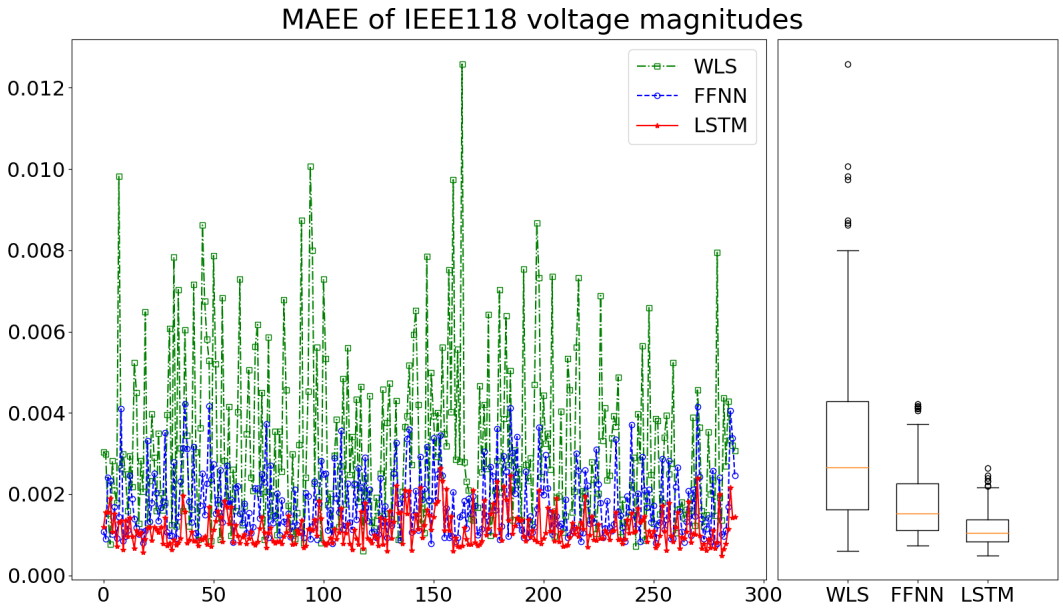

Figure 4.4: MAEEs of voltage magnitudes in IEEE 118-bus system

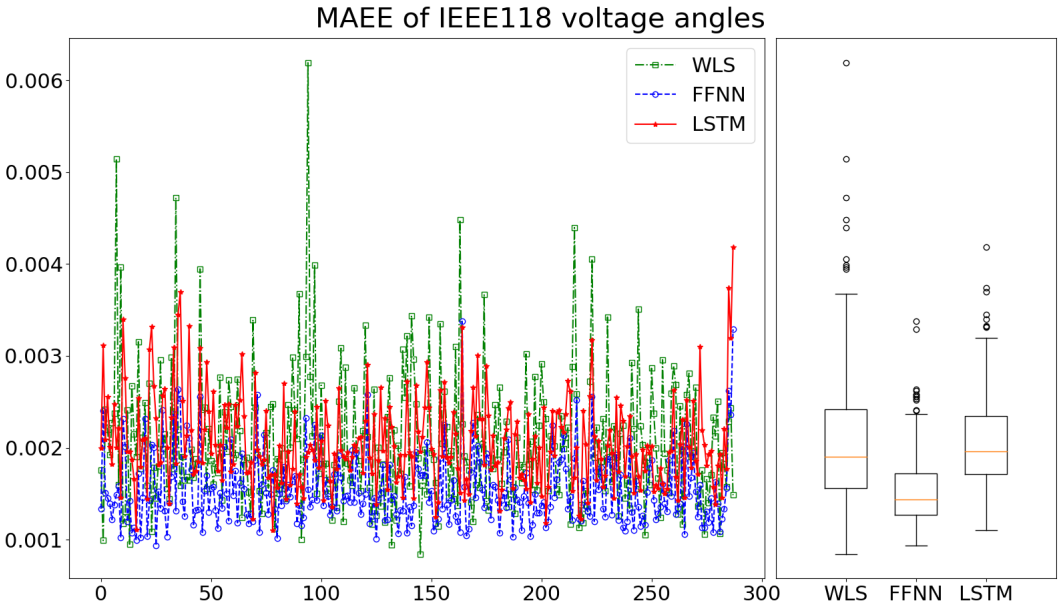

Figure 4.5: MAEEs of voltage angles in IEEE 118-bus system

Table 4.3: Mean and standard deviation of MAEEs in IEEE 118-bus system

IEEE118	Mean(MAEE)		SD(MAEE)	
	V	θ	V	θ
WLS	0.0032	0.0021	0.0021	0.0007
FFNN	0.0018	0.0015	0.0008	0.0004
LSTM	0.0011	0.0021	0.0004	0.0005

Performance Analysis - Robustness

A state estimator is considered statistically robust if the estimated state withstand deviations of measurements [4]. The robustness of PGDL is compared to WLS and the more robust WLAV. It is known that WLAV attempts to find the estimated states by minimizing the L_1 norm between acquired measurements and estimated measurements, which is equivalent to a LP problem [4, 22]. In this thesis, WLAV state estimator is implemented using IPOPT [23].

To test the robustness, 2% - 3% of measurements are randomly selected and replaced with bad data. The results of the last day are used to validate the robustness. Three bad data scenarios are considered:

- *Scenario 1:* The magnitudes of selected measurements are contaminated with 20 times of standard deviation.
- *Scenario 2:* The selected measurements are set to zero, representing missing data in reality.
- *Scenario 3:* The signs of selected measurements are reversed, representing mis-communication in reality.

For each scenario, multiple simulation runs are conducted and each run has 24 hours (288 observations) of single system snapshots.

For the IEEE 14-bus system with 30 measurements, 50 simulation runs are conducted with 2 bad data points randomly selected for each run. Note that both WLS and WLAV have non-converging cases. For example, WLS has 1, 7, and 11 non-converging cases for three scenarios respectively , while WLAV has 1 non-converging case for scenario 1. Note that PGDL methods will always produce results without convergence concerns. We remove non-convergence cases for WLS and WLAV, and report the results in Fig. 4.6. It is observed that LSTM-based model outperforms the others in all three scenarios. The LSTM-based model produces the lowest median in errors and smaller inter-quartile range compared to WLS and WLAV. Surprisingly, FFNN shows the worst performance in all four methods, but note that this is because non-converging cases are not counted in WLS and WLAV.

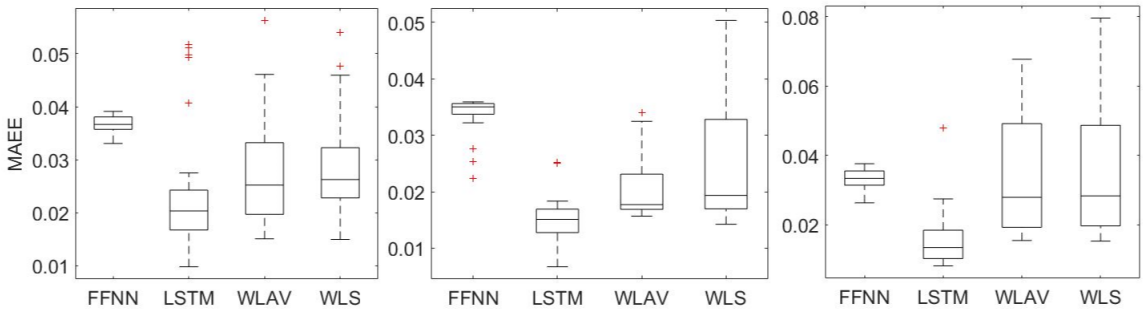

Figure 4.6: MAEE distribution with bad data in IEEE14

For the IEE 118-bus system with 330 measurements, 20 simulation runs are conducted with 5 bad data and 10 bad data randomly selected respectively. The results are reported in Fig. 4.7. One can see that the overall performance of LSTM is reasonably good compared to other methods. In all three scenraios, LSTM has the lowest average errors while WLAV has the lowest error variance.

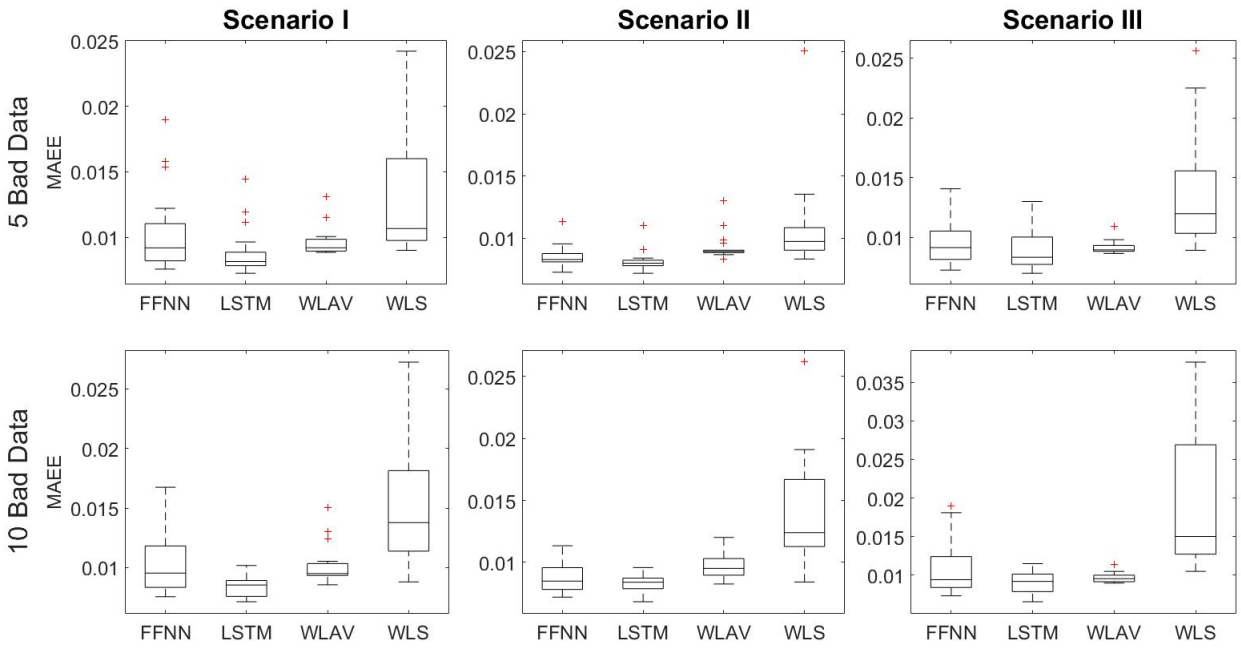

Figure 4.7: MAEE distribution with bad data in IEEE118

Overall, comparing two different DNNs, the mean accuracy improvement of one over the other is subtle given that both methods achieve far better results than WLS. However, LSTM is definitely more stable than FFNN in terms of lower standard deviation of the error index. This can be further seen from the robustness test, where FFNN fails to maintain a low error index given corrupted data, while LSTM remains good accuracy and stable performance.

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

A physics-guided deep learning state estimator is proposed in the paper. The accuracy and robustness are examined in standard IEEE cases. The results show that machine learning based estimator appears to be promising in real control centers. However, it is not intended to replace physics-based methods but rather assist in physics-based state estimator. For instance, it can be used in combination when WLS or WLAV could not yield converged solutions. In addition, incorporating physical models in the machine learning algorithms results in meaningful result for state estimation, and improves both accuracy and robustness. In contrast to snapshot-based estimation by traditional methods, the learning capacity of deep neural networks is fully exploited to model the temporal correlations among system states. Consequently, the proposed PGDL method shows higher accuracy compared to WLS, and is more robust to bad data compared to WLS and WLAV. As for further work, system parameter errors need to be considered.

LIST OF REFERENCES

[1] F. C. Schweppe and J. Wildes, "Power system static-state estimation, part i: Exact model," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-89, no. 1, pp. 120–125, Jan 1970.

[2] F. C. Schweppe and D. B. Rom, "Power system static-state estimation, part ii: Approximate model," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-89, no. 1, pp. 125–130, Jan 1970.

[3] M. Göl and A. Abur, "Lav based robust state estimation for systems measured by pmus," *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 1808–1814, July 2014.

[4] A. Abur and A. Expósito, *Power System State Estimation: Theory and Implementation*. CRC Press, 2004.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[8] S. Fan and L. Chen, "Short-term load forecasting based on an adaptive hybrid method," *IEEE Transactions on Power Systems*, vol. 21, no. 1, pp. 392–401, 2006.

[9] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Transactions on power systems*, vol. 16, no. 1, pp. 44–55, 2001.

[10] W. W. Kotiuga and M. Vidyasagar, "Bad data rejection properties of weughted least absolute value techniques applied to static state estimation," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-101, no. 4, pp. 844–853, April 1982.

[11] J. Currie and D. I. Wilson, "OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User," in *Foundations of Computer-Aided Process Operations*, N. Sahinidis and J. Pinto, Eds., Savannah, Georgia, USA, 8–11 January 2012.

[12] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251 – 257, 1991.

[13] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[15] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, pp. III–1310–III–1318.

[16] X. Li, Y. Grandvalet, and F. Davoine, "Explicit inductive bias for transfer learning with convolutional networks," *CoRR*, vol. abs/1802.01483, 2018. [Online]. Available: http://arxiv.org/abs/1802.01483

[17] L. N. Smith, "No more pesky learning rate guessing games," *CoRR*, vol. abs/1506.01186, 2015.

[18] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[19] Nyiso. [Online]. Available: http://www.nyiso.com

[20] Matpower. [Online]. Available: http://www.pserc.cornell.edu/matpower/#docs

[21] E. E. Fetzer and P. M. Anderson, "Observability in the state estimation of power systems," *IEEE Transactions on Power Apparatus and Systems*, vol. 94, no. 6, pp. 1981–1988, Nov 1975.

[22] L. Mili, M. G. Cheniae, and P. J. Rousseeuw, "Robust state estimation of electric power systems," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 41, no. 5, pp. 349–358, May 1994.

[23] J. Currie and D. I. Wilson, "OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User," in *Foundations of Computer-Aided Process Operations*, N. Sahinidis and J. Pinto, Eds., Savannah, Georgia, USA, 8–11 January 2012.